AMENDMENTS TO THE SPECIFICATION

Please replace paragraph [0006] as follows:

[0006] When we consider programs executed on top of a virtual machine (VM), as it is the case with the JAVA^{TM} platform, we have to mention another alternative to code instrumentation: VM-generated events, or "VM hooks." The VM itself can be instrumented to generate events such as method entry/exit, object allocation, monitor enter, etc. This is done essentially by placing calls to user-suppliable functions in the relevant places in the VM code, for example in the interpreter code executed upon method entry. Some events that are important when profiling a Java application, for example a garbage collection event, cannot be generated using bytecode instrumentation at all. However, for most of the other events, in particular for method entry/exit and object allocation, it has been found over time that their support inside a JVM complicates the latter, sometimes requiring a significant effort from developers, and at run time may cost more than equivalent bytecode instrumentation. This is true at least for VMs intended to be used for general-purpose desktop and server applications, in contrast with those used in cell phones and smaller devices. As a result, it has been recently decided by the expert group established to define a new JVM profiling API, that in the forthcoming specification, many of the VM-generated events, including method entry/exit and object allocation, will be optional and not required to be supported by all conforming JVMs (see JSR 163—Java Platform Profiling Architecture, which was published on the webpage of Java Community Process ~~http://www.jcp.org/jsr/detail/163.jsp, 2002)~~. Bytecode instrumentation is the recommended mechanism for their generation.